Structured Programming Language (CSE113)
Lecture # 02
Operators in C programming

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Assignment Operators
5. Bitwise Operators
6. Misc Operators

1. Arithmetic Operators

int x = 13, y = 6, z;

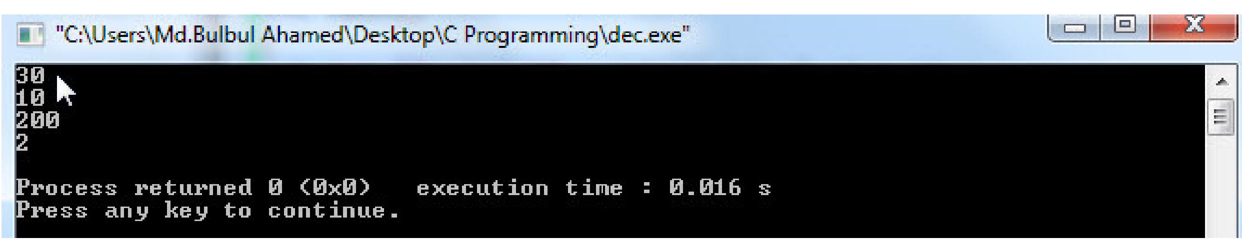
Operator	symbol	Example	Result
Addition	+	z = x + y	19
Subtraction	-	z = x - y	7
Multiplication	*	z = x * y	78
Division	/	z = x / y	2
Modulus	%	z = x % y	1
(Post) increment	++	x++ x = x + 1	13
(Pre) increment	++	++x x = x + 1	14
(Post) Decrement	--	x--	13
(Pre) Decrement	--	--x	12

Program Example 1: Arithmetic Operators

Source Code:

```
#include<stdio.h>
int main()
{
        int x = 20, y = 10;  //Variable Declaration
        printf("%d\n", x + y); //Addition
        printf("%d\n", x - y); //subtraction
        printf("%d\n", x * y); //Multiplication
        printf("%d\n", x / y); //Division
        return 0;
}
```

Output:

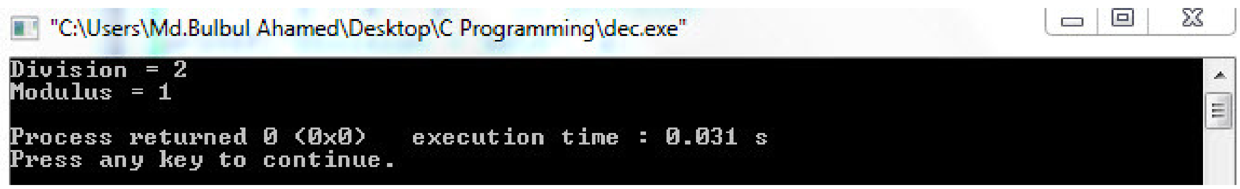


Program Example 2: division and modulus

Source Code:

```
#include<stdio.h>
int main()
{
        int x = 13, y = 6, z;
            z = x / y;   // division
        printf("Division = %d\n", z);
            z = x % y;   // modulus
        printf("Modulus = %d\n", z);
        return 0;
}
```

Output:

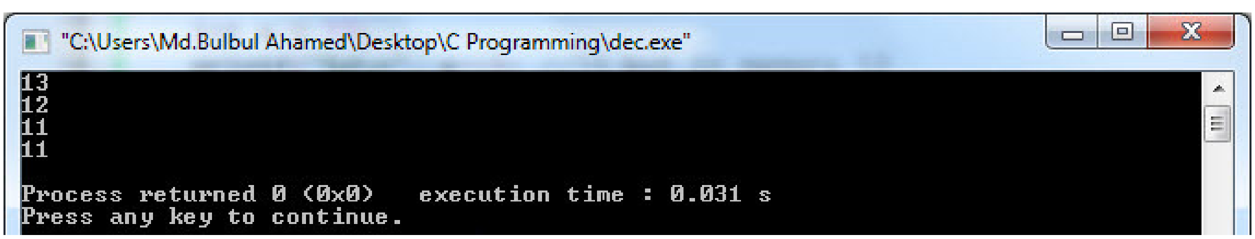


Program Example 3: Pre and Post decrement

Source Code:

```
#include<stdio.h>
int main()
{
  int x = 13;
  printf("%d\n", x--); //13 but in memory 12
  printf("%d\n", x); //12
  printf("%d\n", --x); //11
  printf("%d\n", x);//11
  return 0;
}
```

Output:



2. Relational Operators

int x = 13, y = 6;

Operator	symbol	Example	Result
Equal	==	x == y 13 == 6	0
Not equal	!=	x != y 13 != 6	1
Greater than	>	x > y 13 > 6	1
Less than	<	x < y 13 < 6	0
Greater than or equal to	>=	x >= y 13 >= 6 6 >= 6	1
Less than or equal to	<=	x <= y 13 <= 6	0

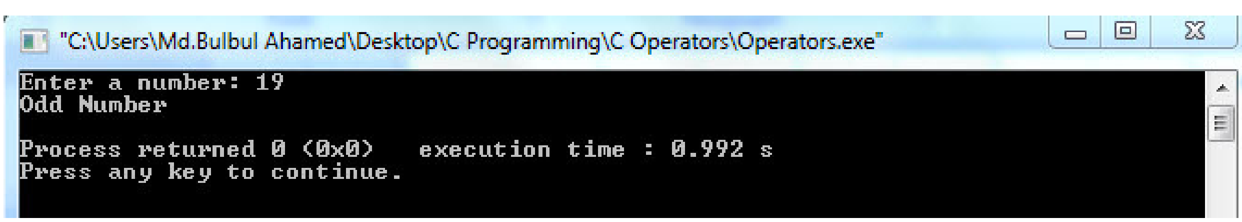
Program Examples: Relational Operators

Q1. Write a C program to check a number whether it is odd or even.

Source Code:

```
#include<stdio.h>
int main()
{
   int number;
   printf("Enter a number: ");
   scanf("%d",&number);
   if(number%2 == 0)
      printf("Even Number\n");
   else
      printf("Odd Number\n");
   return 0;
}
```

Output:

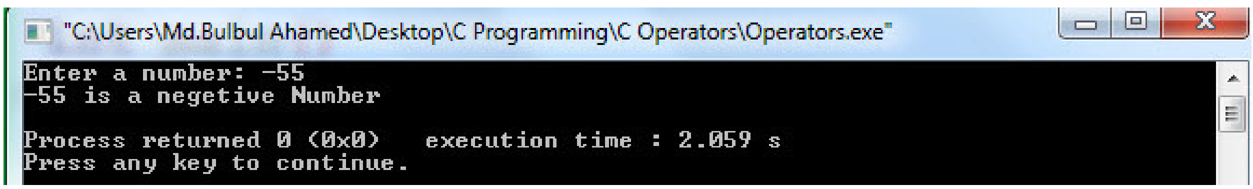


Q2. Write a C program to check a number whether it is positive or negative.

Source Code:

```
#include<stdio.h>
int main()
{
   int number;
   printf("Enter a number: ");
   scanf("%d",&number);
   if(number < 0)
      printf("%d is a negetive Number\n",number);
   else
      printf("%d is a postive Number\n",number);
   return 0;
}
```

Output:

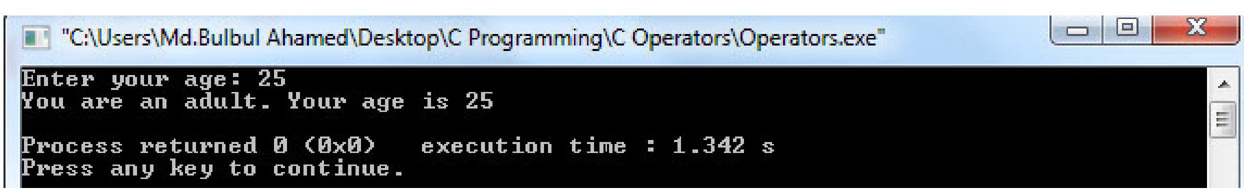


Q3. Write a C program to display You are an adult if the value of age is 18 or more, otherwise it will display You are a child.

Source Code:

```
#include<stdio.h>
int main()
{
   int age;
   printf("Enter your age: ");
   scanf("%d",&age);
   if(age >= 18)
      printf("You are an adult. Your age is %d\n",age);
   else
      printf("You are a child. Your age is %d\n",age);
   return 0;
}
```

Output:



3. Logical Operators

Operator	symbol
Logical AND	&&
Logical OR	||
Logical NOT	!

Q1. Write a C program to find the grade according to the following table:

Marks (M)	Grade
80 – 100	A+
70 – 79	A
60 – 69	A-
50 – 59	B
40 –49	C
33 – 39	D
0 – 32	F

Here, the value of **marks(M)** must be taken from user.

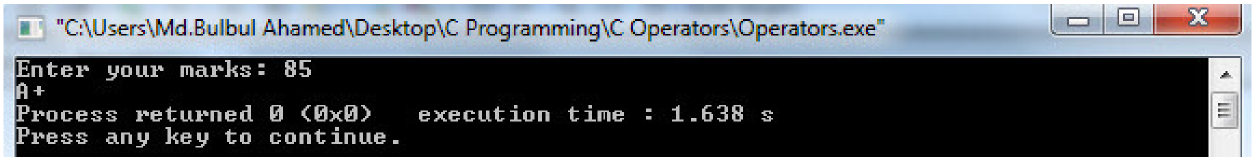
N.B. If any user put **more than 100** or **less than 0** the program will give the following error message:

You entered invalid marks!

<u>Source Code:</u>

```
#include<stdio.h>
int main()
{
	int M; //M : marks
	printf("Enter your marks: ");
	scanf("%d", &M);
	if(M > 100 || M < 0) //double pipe
		printf(" You Entered Invalid Marks! ");
	else if(M >= 80)
		printf("A+");
	else if(M >= 70)
		printf("A");
	else if(M >= 60)
		printf("A-");
	else if(M >= 50)
		printf("B");
	else if(M >= 40)
		printf("C");
	else if(M >= 33)
		printf("D");
	else
		printf("F");
	return 0;
}
```

Output:



4. Assignment Operators

The following table lists the assignment operators supported by the C language –

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand	C = A + B will assign the value of A + B to C
+=	Add AND assignment operator. It adds the right operand to the left operand and assign the result to the left operand.	C += A is equivalent to C = C + A
−=	Subtract AND assignment operator. It subtracts the right operand from the left operand and assigns the result to the left operand.	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator. It multiplies the right operand with the left operand and assigns the result to the left operand.	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator. It divides the left operand with the right operand and assigns the result to the left operand.	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator. It takes modulus using two operands and assigns the result to the left operand.	C %= A is equivalent to C = C % A

5. Bitwise Operators

Bitwise operator works on bits and perform bit-by-bit operation. The following table lists the bitwise operators supported by C. Assume variable 'A' holds 60 and variable 'B' holds 13, then −

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) = 12, i.e., 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) = 61, i.e., 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) = 49, i.e., 0011 0001
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) = -61, i.e,. 1100 0011 in 2's complement form.
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 = 240 i.e., 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 = 15 i.e., 0000 1111

6. Misc Operators

Besides the operators discussed above, there are a few other important operators including **sizeof** and **? :** supported by the C Language.

Operator	**Description**	**Example**
sizeof()	Returns the size of a variable.	sizeof(a), where a is integer, will return 4.
&	Returns the address of a variable.	&a; returns the actual address of the variable.
*	Pointer to a variable.	*a;
? :	Conditional Expression.	If Condition is true ? then value X : otherwise value Y

Q1. Write a C program using conditional expression (ternary operator) to check a number whether it is odd or even.

Source Code:

```
#include<stdio.h>
int main()
{
   int n;
   printf("Enter a numebr : ");
   scanf("%d",&n);
   (n%2==0)? printf("Even Number\n") : printf("Odd Number\n");
   return 0;
}
```

Output:

